Application Report
SPRA492A

Interrupt Handling Using Extended Addressing of the TMS320C54x Family

Chuck Brokish *Field Design & Applications*

Abstract

Special consideration should be given to handling interrupts when using extended addressing on the Texas Instruments (TI™) TMS320C54x family of digital signal processing devices. Because extended addressing uses up to 23 address lines on a 16-bit device, the user must account for saving and restoring the extended address values. This operation is handled *automatically* when using the far instructions. However, when dealing with interrupts, no assumptions can be made on the part of the architecture as to whether you are actually utilizing the extended memory capabilities of the device. Therefore, only the 16-bit “local” address is automatically saved on the stack on acknowledging an interrupt.

This document covers available options for handling interrupt service routines (ISRs) when using the extended addressing capability of the TMS320C54x. Included in this document are code examples and memory configuration options.

Contents

Tables

Introduction

The ‘C54x is a 16-bit device. As such, all data paths and memory are configured to operate on 16 bits of information. There can be up to 23 bits of significant program address information when using the extended addressing capabilities of the C54x.[1] This means that calls and returns crossing 64K boundaries (16-bit address), also known as *far calls*, require both the program counter (PC) and the extended program counter (XPC) to be pushed onto the stack. This also implies that both elements must be popped off of the stack on returning from a far address.

On interrupts, the C54x does not automatically save both of these values because of the inefficiency it would add to cases in which the extended addressing is not used. Such cases tend to be the most common.

Determining System Requirements

When developing a DSP system, it is necessary to determine the memory requirements. All of the C54x devices support 64K words of address space in program, data, and I/O space. This yields a total reachable memory space of 192K words without the use of extended addressing. When designing systems that fit these constraints, it is not necessary to use extended addressing; the operation of the system on interrupts handles all saving and restoring of the program counter without issue.

For systems that have large program requirements, it may be necessary to use more than the 64K words of base program address range. In these cases, extended addressing can support up to 23 bits (8M words) of addressable program space. When supporting interrupts within systems using extended addressing, several options are available to handle the saving and restoring of the entire program address.

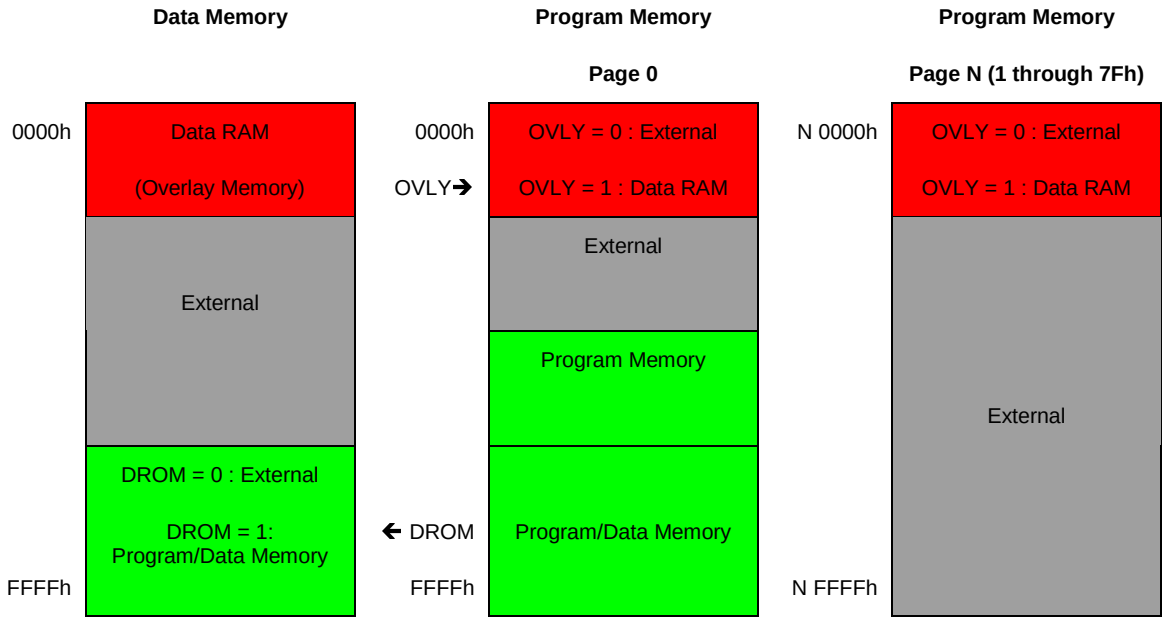
Memory Map of C54x Devices Using Extended Addressing

Several options are available for configuring the memory map. These configurations can be affected by external pin settings at RESET or by internal status register settings. Table 1 shows the memory map options of the C54x devices supporting extended addressing. The boundary addresses of each block of memory vary, depending on the product number and the memory mix available on that part.

Note that the same RAM shown here in data overlay space is available on all pages of extended program memory when the OVLY status bit is set. Internal program memory may be available on page 1, 2, or 3 of custom devices and standard devices with larger internal memory. But the overlay memory remains visible on all program memory pages, if OVLY is set to 1.

[1] Extended addressing is not supported on all TMS320C54x devices.

Table 1. Memory Map Configurations[2] of the C54x



Default Settings

Interrupt Vector Table

When the C54x DSP is released from RESET, the default boot location address is 0FF80h. The vector table is 128 words (80h) long and fills the end of the 64K-word boundary. The vector table of the C54x is relocatable to any 80h-word boundary.[3]

The RESET vector is the first vector within the vector table. The table is comprised of 32 interrupt vector locations—each location being allocated four program words within the table. This allows you to either perform a simple operation and return to your application or branch to the ISR. Branching to the ISR is the most common approach because most ISRs require too much processing to fit within the 4-word constraint of the vector table.

The vector table can be located to another location by changing the interrupt pointer (IPTR) value in the processor mode status register (PMST). IPTR is a 9-bit field that indicates the nine most significant bits (Address[15:7]) of the interrupt vector table.

You are also responsible for making sure that the appropriate program code is located at this new address. This can be accomplished by either of the following:

- Create a separate vector table and place it into the ROM code at the new address.
- Copy the existing vector table to that new address before changing the value of IPTR.

[2] The memory map assumes internal memory is being used (MP/MC = 0).

[3] The vector table must be located within the first 64K words of program space (0000h–FFFFh).

Interrupt Operation of the C54x Processor

The processor takes the following actions at the start of an interrupt of any form (internal, external, software):

Table 2. CPU Actions on Interrupt

CPU Action	Description
1 → INTM	Disable global interrupts.
PC → --*(SP)	Push PC onto predecremented stack.
(IPTR)<<7 + (Vector[n])<<2 → PC	Load PC with address within the vector table.

NOTES:

1) The processor saves the PC to the stack so that, at the end of the ISR, you can simply return to the application at the address where you left off. The new value assigned to the PC is a 16-bit value comprised of the starting address of the vector table and the appropriate location within the vector table for the specific interrupt.
2) The XPC is not saved to the stack, and the new PC value for the vector table is only 16 bits. This means that the vector table must be within the 64K-word program page in which the application is running.

Far Instructions

C54x devices offering extended addressing capability include the special instructions shown in Table 3. These instructions enable the use of the additional program memory.

Table 3. Far Instructions

Instruction	Description
FB[D]	Far branch to address specified
FBACC[D]	Far branch to address contained in Acc[22:0]
FCALL[D]	Far call to address specified
FCALA[D]	Far call to address contained in Acc[22:0]
FRET[D]	Far return
FRETE[D]	Far return and enable global interrupts

Each instruction utilizes the 23-bit program address comprised of the 16-bit PC and the 7-bit XPC. The far branch instructions simply replace the contents of the PC and XPC with the value specified in the instruction or in the accumulator. The far call instructions not only replace the 23-bit program address but also store the old value onto the stack, using two stack locations to do so. The far return Instructions pop both the XPC and the PC from the stack to create the new program address.

The operation of the CPU on a far call is to push both the PC and the XPC onto the stack in that order. In contrast, the call instruction pushes only the PC onto the stack. Likewise, on the occurrence of an interrupt, the CPU pushes only the PC onto the stack.

Operating From Extended Program Memory

For any application software running all of the program flow within page 0, there is never a need to change XPC. Therefore, the interrupt operation of the CPU fully supports the system needs. Table 4 shows an example of a typical code segment from the vector table and the end of the ISR.

For applications using extended addressing, it is necessary to make sure that the vector table can be reached when an unmasked interrupt occurs, regardless which program address the application is running.

Several options are available to enable proper operation of the interrupts while enabling operation from extended program memory. The following sections discuss a few of the options.

Place Un-Interruptible Code in Extended Memory

Many applications include sections of code that are time critical and therefore cannot be interrupted once started. By placing these sections of code in extended memory and placing the interrupt vector table and ISRs in page 0, the vector table and ISRs are always available locally when an interrupt is acknowledged. On return to the interruptible portions of code on page 0, interrupts can be re-enabled, and you can alleviate any concerns over the location of the interrupt vector table and the ISRs.

Make All Interrupt Service Routines Far Callable

To minimize the memory consumption within the user application, it is beneficial to make the fewest possible copies of code sections. However, when creating re-locatable code, there is a built-in understanding that the code will be copied to at least one more location. Such is the case with a re-locatable vector table.

By copying the vector table from its original location at RESET (0FF80h–0FFFFh), you create another copy of the same code. However, by copying the vector table to the overlay memory in data space and setting OVLY to 1, you can access the vector table from overlay memory in program space. This memory is available on *all* pages of extended program memory.

Having the vector table available in overlay memory allows the vector table to be accessible to the CPU regardless from which extended page the program runs.

When making the vector table available on all program pages, ensure that the address to which you intend to branch for the ISR is reachable. Therefore, within the vector table, you must perform a far branch to the ISR and make sure to store the current XPC so that you can return to the proper program page when the ISR is completed.

When completed with the ISR, you may be on a different page in program memory than before the interrupt occurred. Therefore, the return from the ISR should be a far return. Table 5 shows a code example of code in which the ISR has been made far callable and the vector table preserves the XPC. Note that this implies all ISRs should be made far callable.

Note also that the interrupt latency is increased by two cycles with this implementation because you cannot use the delayed version of the far branch instruction (FBD). This is because you must push the XPC to the stack *before* implementing the far branch to avoid changes in the XPC prior to its being saved on the stack.

Table 4. Typical Interrupt Code When NOT USING Extended Memory

Vector Table	Interrupt Service Routine

`IntVec_n: BD ISR_n`
`PSHM ST0`
`PHSM ST1`
`IntVec_m: BD ISR_m`
`PSHM ST0`
`PHSM ST1`
	`ISR_n: PSHM … ;context save`
	`:`
	`<perform ISR>`
	`:`
	`POPM … ;context restore`
	`POPM ST1`
	`POPM ST0`
	`RETE ;return & enable INTs`

Table 5. Typical Interrupt Code When USING Extended Memory

Vector Table	Interrupt Service Routine

`IntVec_n: PSHM XPC`
`PHSM ST0`
`FB ISR_n`
`IntVec_m: PSHM XPC`
`PHSM ST0`
`FB ISR_m`
	`ISR_n: PSHM … ;context save`
	`:`
	`<perform ISR>`
	`:`
	`POPM … ;context restore`
	`POPM ST0`
	`FRETE ;return & enable INTs`

Place Vector Table and All Interrupt Service Routines in Overlay Memory

Another option to making all of the ISRs far callable is to place them in the overlay memory of the C54x. For devices such as the C5410, which has 32K words of RAM in the overlay space, it is likely that not all of this will be needed for data use. In this case, it may be possible to place the ISRs as well as the vector table in overlay memory.

Placing both the vector table and the ISRs in overlay memory makes them available from all pages of extended memory. In this case, neither the vector table nor the ISR must perform far addressing. This allows all interrupts to change the PC to the vector table in the overlay memory, regardless of the current extended program page. The DSP can then vector off to the ISR in the overlay memory and return locally.

TI Contact Numbers

INTERNET

TI Semiconductor Home Page
www.ti.com/sc

TI Distributors
www.ti.com/sc/docs/distmenu.htm

PRODUCT INFORMATION CENTERS

Americas

Phone	+1(972) 644-5580
Fax	+1(972) 480-7800
Email	sc-infomaster@ti.com

Europe, Middle East, and Africa

Phone	
Deutsch	+49-(0) 8161 80 3311
English	+44-(0) 1604 66 3399
Español	+34-(0) 90 23 54 0 28
Francais	+33-(0) 1-30 70 11 64
Italiano	+33-(0) 1-30 70 11 67
Fax	+44-(0) 1604 66 33 34
Email	epic@ti.com

Japan

Phone	
International	+81-3-3344-5311
Domestic	0120-81-0026
Fax	
International	+81-3-3344-5317
Domestic	0120-81-0036
Email	pic-japan@ti.com

Asia

Phone	
International	+886-2-23786800
Domestic	
Australia	1-800-881-011
TI Number	-800-800-1450
China	10810
TI Number	-800-800-1450
Hong Kong	800-96-1111
TI Number	-800-800-1450
India	000-117
TI Number	-800-800-1450
Indonesia	001-801-10
TI Number	-800-800-1450
Korea	080-551-2804
Malaysia	1-800-800-011
TI Number	-800-800-1450
New Zealand	000-911
TI Number	-800-800-1450
Philippines	105-11
TI Number	-800-800-1450
Singapore	800-0111-111
TI Number	-800-800-1450
Taiwan	080-006800
Thailand	0019-991-1111
TI Number	-800-800-1450
Fax	886-2-2378-6808
Email	tiasia@ti.com

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE (“CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty, or endorsement thereof.